Machine Learning Major Project

Salary Prediction Using Regression Models

Submitted By: Somya Sharma

Problem Statement:

Business Problem: To ensure there is no discrimination between employees, it is imperative for the Human Resources department of Company X to maintain a salary range for each employee with a similar profile. Apart from the existing salary, a considerable number of factors, such as an employee's experience and other abilities, are evaluated during interviews. Given the data related to individuals who applied to Company X, models can be built that automatically determine the salary to be offered if a prospective candidate is selected. This model seeks to minimize human judgment in salary decisions.

Goal & Objective:

The objective of this exercise is to build a model, using historical data, that will determine the salary to be offered to an employee, minimizing manual judgment in the selection process. The approach aims to be robust and eliminate any discrimination in salary among employees with similar profiles.

Primary Objective:

Develop a machine learning model using historical employee and application data to predict the appropriate salary offer for new candidates.

- Minimize manual and subjective judgments in salary settings.
- Create a fair, data-driven framework for offers.
- Address equity by guaranteeing that employees with similar profiles receive comparable salaries.

➔ Exploratory Data analysis (EDA)

The dataset contains 25000 rows and 29 columns . The Id and Application_ID columns were dropped as they were irrelevant in model training.

- **Dataset Overview**
  - Loaded the dataset using pandas.
  - Checked shape of dataset → number of rows & columns.
  - Previewed data using df.head().
  - Checked data types and memory usage with df.info().
  - Summarized statistics of all columns with df.describe(include='all').
- **Missing & Duplicate Values**
  - Checked for null values using df.isnull().sum().
  - Verified if dataset contained duplicate records.
- **Univariate Analysis**
  - Plotted **distribution plots** for numerical columns (using sns.histplot with KDE).
  - Plotted **frequency distribution** for categorical columns.
- **Bivariate Analysis**
  - Visualized relationship between **numerical features and Expected_CTC** using scatter/rel plots.
  - Visualized relationship between **categorical features and Expected_CTC** using categorical plots (bar/box).
- **Multivariate Analysis**
  - Created **pair plots** (sns.pairplot) to study relationships across features.
  - Plotted **boxplots** of numerical features to check spread & outliers.

- Used **heatmaps** (sns.heatmap) to study correlation among numerical features.

- **Feature Encoding & Transformation**
  - Encoded categorical variables into numerical form.
  - Created df_encoded.head() to check transformed dataset.
- **Dimensionality Reduction (PCA)**
  - Applied **PCA (Principal Component Analysis)** for feature reduction.
  - Checked explained variance ratio of components.
  - Visualized PCA-transformed feature correlations using heatmap.
  - Evaluated regression performance on PCA-transformed data.

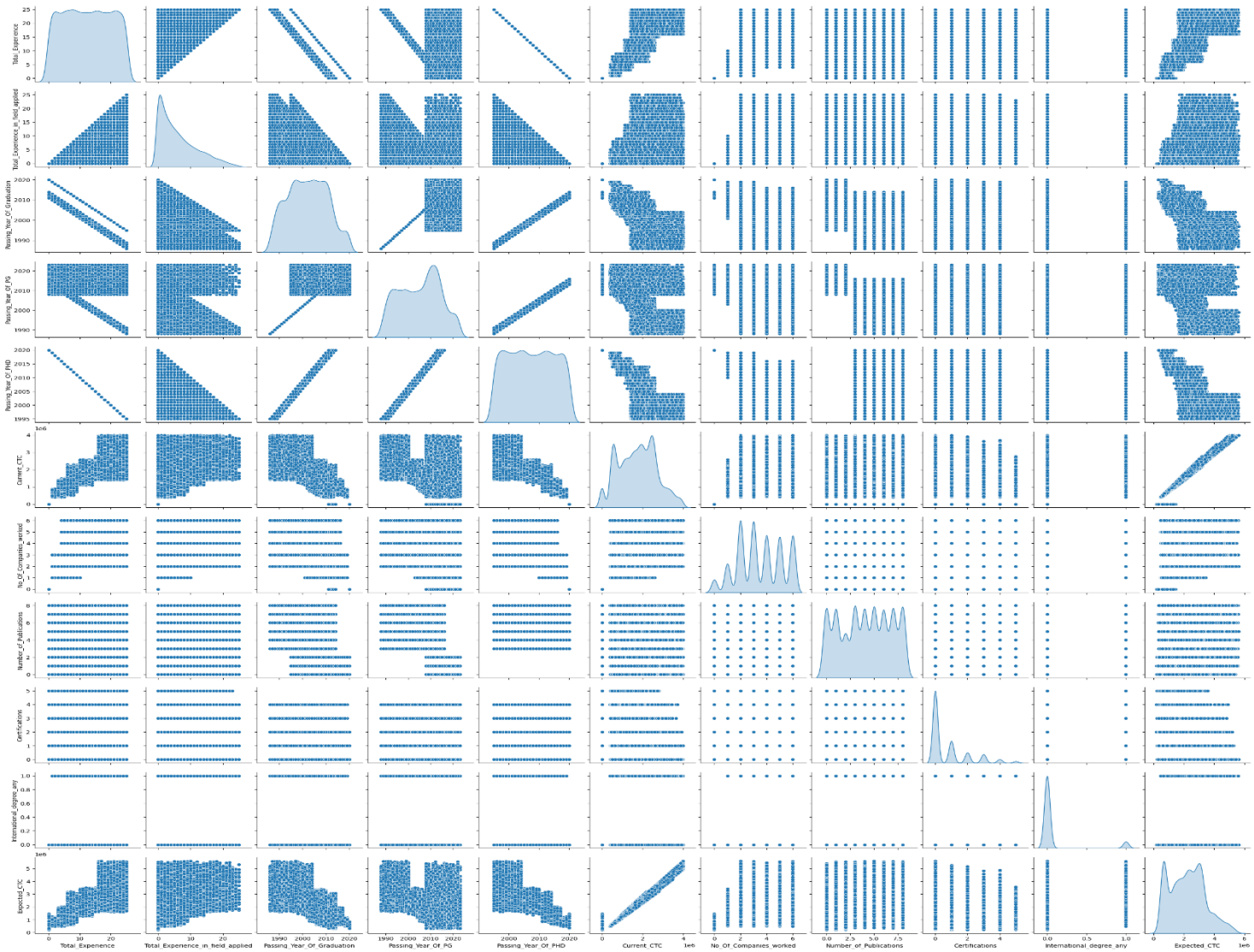

Fig 1.1: Pair plot showing relationships between selected features

➔ Data Cleaning and Preprocessing

- Outliers were inspected visually and found to be minimal; hence they were retained . (fig 1.2)
- The correlation matrix reveled high collinearity between variables like Passing_Year_Of_PHD , PG and Graduation which were therefore dropped.
- Missing values were replaced with “NA” for categorical variables , allowing One-Hot Encoding to handle them explicitly.
- Numerical variables were standardized using StandardScaler( ) from scikit-learn.

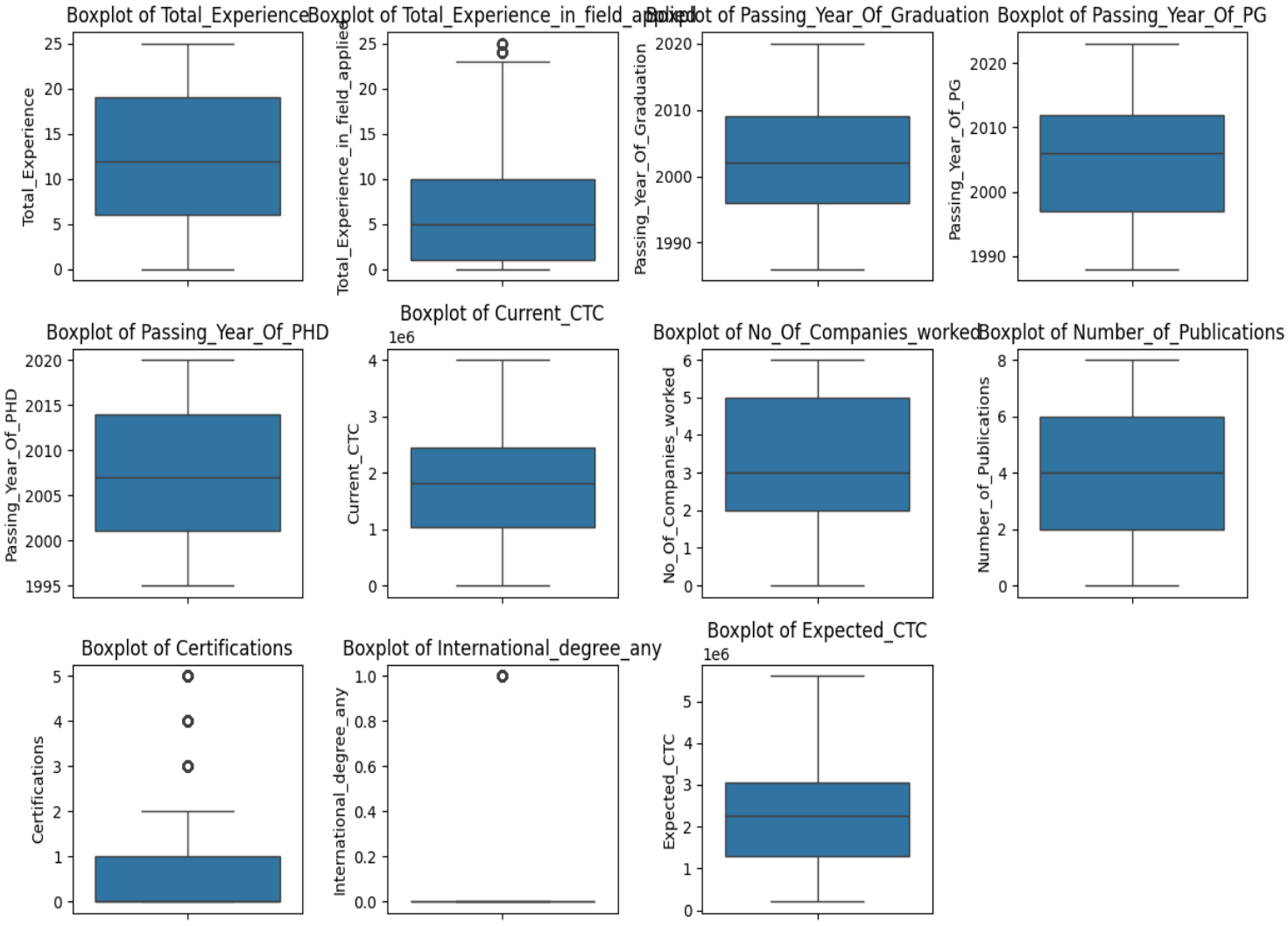

Fig 1.2: Box Plot showing the outliers

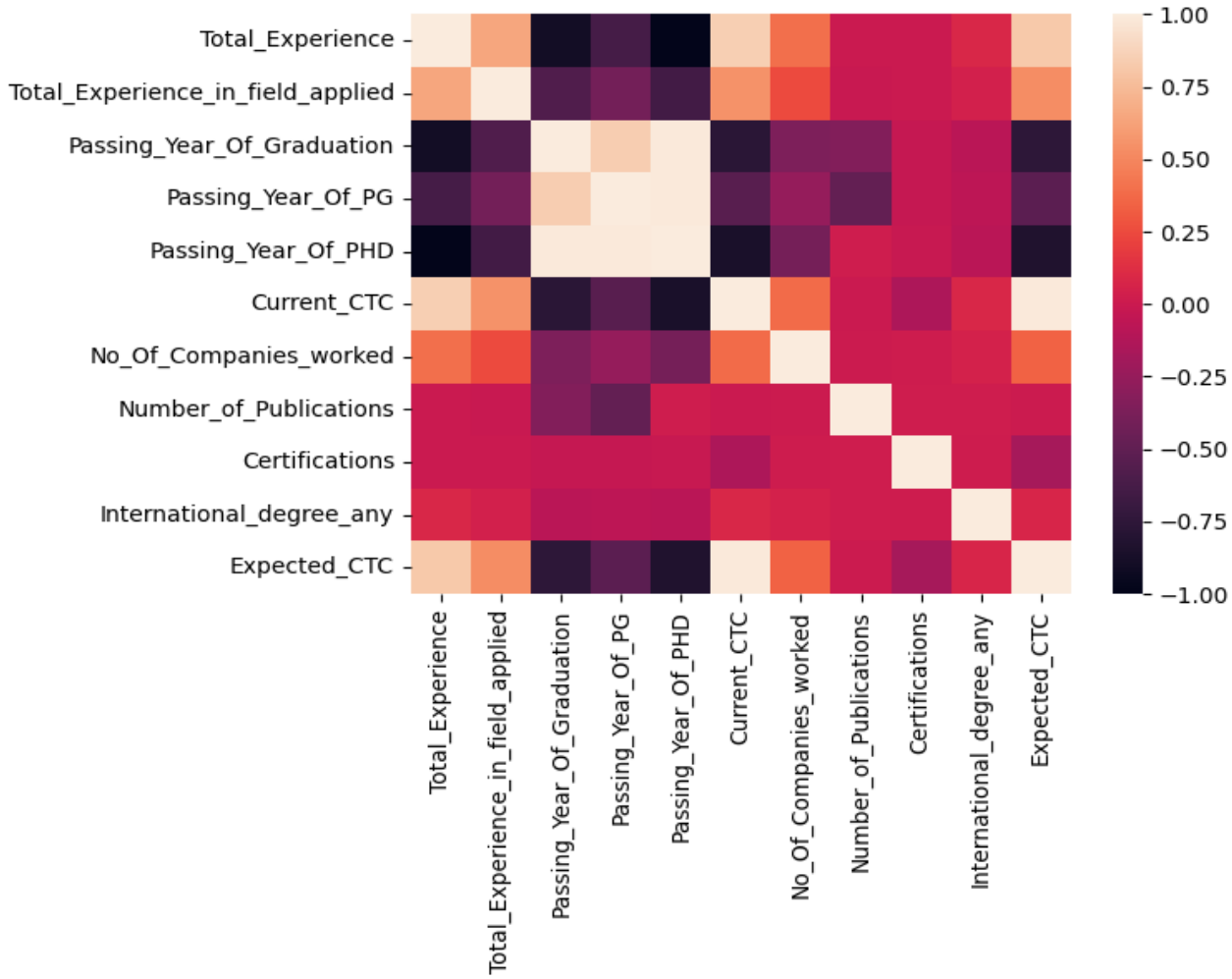

Fig 1.3: Heatmap of Feature Correlation

Model Used for Training the Dataset:

1. **Linear Regression**
2. **Linear Regression with Principal Component Analysis (PCA)**
3. **Decision Tree Regression**
4. **Random Forest Regression**
5. **XGBoost**

□ Key Insights from Dtypes + Unique Counts

🔢 Numerical Columns

- int64 and float64 types like Total_Experience, Current_CTC, Expected_CTC are likely continuous or ordinal.
- Columns like No_Of_Companies_worked, Certifications, Number_of_Publications have **low cardinality**, so they can be treated as categorical or binned.

□ Categorical Columns

- object types like Department, Role, Industry, Organization, etc., have moderate cardinality (11–24 unique values).
- Education has only 4 unique values — perfect for one-hot encoding or label encoding.
- Inhand_Offer and International_degree_any are binary — treat as boolean or 0/1.

□ Identifiers

- IDX and Applicant_ID have high uniqueness — likely row IDs or primary keys. These should **not** be used as features.

🔍 What This Tells You

- **Multiple salary clusters**: You might be seeing entry-level, mid-career, and senior-level applicants.
- **Skewed distribution**: There's a long tail toward higher CTCs, so consider log-transforming Expected_CTC if you use linear models.
- **Outliers**: If you spot extreme values (e.g., > ₹5M), you might want to cap or winsorize them to stabilize training.